DOCUMENT RESUME

ED 403 310 TM 026 042

AUTHOR Haertel, Edward H.
TITLE Software for the Application of Discrete Latent Structure Models to Item Response Data.
SPONS AGENCY National Science Foundation, Arlington, VA.
PUB DATE Jul 96
NOTE 42p.; For a document describing the use of these programs, "Latent Traits or Latent States? The Role of Discrete Models for Ability and Performance," see TM 026 041.
PUB TYPE Computer Programs (101)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Ability; *Academic Achievement; *Computer Software; *Educational Assessment; *Mathematical Models; Programming Languages; *Science Education
IDENTIFIERS *FORTRAN Programming Language; Mapping

ABSTRACT

These FORTRAN programs and MATHEMATICA routines were developed in the course of a research project titled "Achievement and Assessment in School Science: Modeling and Mapping Ability and Performance." Their use is described in other publications from that project, including "Latent Traits or Latent States? The Role of Discrete Models for Ability and Performance." Four FORTRAN programs (CLOSURE, DISATT, INTERSECT, and SUPERCLOSE) and one set of MATHEMATICA routines (Ability/Task Representations) are included. The use of each program is described in the comments included in the source code listings. (Author/SLD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

TM

ED 403 310

Software for the Application of Discrete Latent Structure Models to Item Response Data

Edward H. Haertel
Stanford University
July 1996

These FORTRAN programs and MATHEMATICA routines were developed in the course of a research project titled "Achievement and Assessment in School Science: Modeling and Mapping Ability and Performance," sponsored by the National Science Foundation (Award No. 9154527, Edward H. Haertel, Principal Investigator). Their use is described in other publications of that project, including "Latent Traits or Latent States? The Role of Discrete Models for Ability and Performance," which is also available as an ERIC document.

Four FORTRAN programs (CLOSURE, DISATT, INTERSECT, and SUPERCLOSE) and one set of MATHEMATICA routines (Ability/Task Representations) are included. The use of each program is described in the comments included in the source code listings.

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

EDWARD HAERTEL

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

TM026042

BEST COPY AVAILABLE

```
C     CLOSUREV2
C
C     PROGRAM TO MAP NUMBER OF LRPS IN CLOSED LATTICE AS FUNCTION OF LRPS
C     INPUT.  PROGRAM REQUESTS NAME OF FILE CONTAINING (USUALLY) LIST OF
C     HIGH-FREQUENCY MANIFEST PATTERNS.  LIMIT OF 30 ELEMENTS IN PATTERN.
C     STORAGE OVERFLOW WILL BE DETECTED.
C
C            ***************************************************
C            *                                                 *
C            *      AUTHOR:  EDWARD H. HAERTEL                 *
C            *               STANFORD UNIVERSITY               *
C            *               STANFORD, CA  94305-3096          *
C            *               HAERTEL@LELAND.STANFORD.EDU       *
C            *               415-725-1251                      *
C            *               415-725-7412 (FAX)                *
C            *                                                 *
C            *     PERMISSION IS GRANTED TO USE THIS PROGRAM   *
C            *     FOR NONCOMMERCIAL PURPOSES PROVIDED THE     *
C            *     ORIGINAL AUTHOR IS CREDITED.  NO WARRANTY   *
C            *     AS TO THE ACCURACY OR UTILITY OF THE        *
C            *     PROGRAM FOR ANY PURPOSE IS EXPRESSED        *
C            *     OR IMPLIED.  PERMISSION IS ALSO GRANTED     *
C            *     TO MODIFY THE PROGRAM AND/OR TO INCORPORATE *
C            *     THIS CODE OR ALGORITHMS INTO OTHER          *
C            *     PROGRAMS FOR NONCOMMERCIAL PURPOSES.        *
C            *                                                 *
C            ***************************************************
C
C     MODIFIED 3/26/92 TO PROVIDE ADDITIONAL OUTPUT AND ANALYSIS
C
      INTEGER LIST(2,5000),TEMP(3,1000),IN(30)
      REAL*8 FREQ,CFREQ,RATIO
      CHARACTER*80 FILENAME,FORMAT
      CHARACTER*40 BLANK
      LOGICAL EXIST
      COMMON/START/ISTART
      DATA BLANK/'                                        '/
      WRITE(9,80)'NUMBER OF ITEMS IN EACH PATTERN'
      READ(9,81)NI
   80 FORMAT(A)
   81 FORMAT(I20)
      ISTART=2**(NI-1)
      WRITE(9,80)'NUMBER OF PATTERNS TO BE READ'
      READ(9,81)MAXIN
      ISER=MAXIN+1
    2 WRITE(9,80)'INPUT FILENAME'
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(EXIST)GO TO 3
      WRITE(9,80)'FILE NOT FOUND'
      GO TO 2
    3 OPEN(11,FILE=FILENAME)
      WRITE(9,80)'INPUT DATA FORMAT FOR FREQUENCY (F FORMAT)'
      WRITE(9,80)'FOLLOWED BY PATTERN VECTOR (I FORMAT)'
      READ(9,80)FORMAT
      WRITE(9,80)'OUTPUT FILENAME FOR FUNCTION VALUES AND STATS'
      READ(9,80)FILENAME
      OPEN(12,FILE=FILENAME)
```

```
      WRITE(9,80)'OUTPUT FILENAME FOR COMPLETED LRP LIST'
      READ(9,80)FILENAME
      OPEN(13,FILE=FILENAME)
C     INITIALIZE AND INPUT FIRST CASE
      NIN=1
      NOUT=1
      READ(11,FORMAT)FREQ,(IN(I),I=1,NI)
      CFREQ=FREQ
      RATIO=1.D0
      WRITE(9,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
      WRITE(12,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
   82 FORMAT(30I1)
C**** THE FOLLOWING FORMAT IS INCONSISTENT WITH PROGRAM SPECS ABOVE.  IF
C**** NUMBER OF ITEMS IS GREATER THAT ABOUT 10, CHANGE 'T40' TO 'T54' IN
C**** FORMAT 83
   83 FORMAT(3I5,T40,F7.3,F9.3,F6.3,T21,30I1)
C****
      LIST(1,1)=1
      LIST(2,1)=IPACK(IN,NI)
C     HERE TO INPUT NEXT VALUE
    1 IF(NIN.GE.MAXIN)GO TO 40
      NIN=NIN+1
      READ(11,FORMAT)FREQ,(IN(I),I=1,NI)
      CFREQ=CFREQ+FREQ
      NEW=IPACK(IN,NI)
C     CHECK IF NEW IS ALREADY IN THE LIST, AND STORE IN TEMP IF NOT
      CALL SEARCH(NEW,LIST,NOUT,IRES,IPT)
      IF(IRES.GT.0)GO TO 5
C     NEW PATTERN HAS ALREADY BEEN ADDED--REPLACE SERIAL NUMBER
      LIST(1,IPT)=NIN
      RATIO=DFLOAT(NOUT)/DFLOAT(NIN)
      WRITE(9,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
      WRITE(12,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
      GO TO 1
    5 TEMP(1,1)=NEW
      TEMP(2,1)=NIN
      TEMP(3,1)=IPT
      LAST=1
C     CHECK INTERSECTIONS WITH ALL PATTERNS ALREADY IN LIST
      DO 10 I=1,NOUT
      J=INTER(NEW,LIST(2,I))
      CALL SEARCH(J,LIST,NOUT,IRES,IPT)
      IF(IRES.EQ.0)GO TO 10
C     HERE TO ADD NEW PATTERN TO LIST
      LAST=LAST+1
      TEMP(1,LAST)=J
      TEMP(2,LAST)=ISER
      TEMP(3,LAST)=IPT
      ISER=ISER+1
   10 CONTINUE
C     SORT ADDITIONAL PATTERNS, ELIMINATE DUPLICATES, AND UPDATE LIST
      KEEP=1
      IF(LAST.EQ.1)GO TO 16
      CALL SORT(TEMP,LAST,3,3)
      DO 15 I=2,LAST
      IF(TEMP(1,I-1).NE.TEMP(1,I))GO TO 12
      TEMP(3,I)=-1
      GO TO 15
```

```
   12 TEMP(3,I)=TEMP(3,I)+KEEP
      KEEP=KEEP+1
   15 CONTINUE
   16 IEND1=NOUT
      NOUT=NOUT+KEEP
      IEND=NOUT
      RATIO=DFLOAT(NOUT)/DFLOAT(NIN)
      WRITE(9,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
      WRITE(12,83)NIN,NOUT,NOUT-NIN,FREQ,CFREQ,RATIO,(IN(I),I=1,NI)
   17 IF(TEMP(3,LAST).GT.0)GO TO 18
      LAST=LAST-1
      GO TO 17
   18 IF(IEND-TEMP(3,LAST))20,25,30
   20 WRITE(9,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION ABORTED (1)'
      PAUSE
      STOP
   25 LIST(1,IEND)=TEMP(2,LAST)
      LIST(2,IEND)=TEMP(1,LAST)
      IEND=IEND-1
      LAST=LAST-1
      IF(LAST.GT.0)GO TO 17
      IF(IEND.EQ.IEND1)GO TO 1
      WRITE(9,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION ABORTED (2)'
      PAUSE
      STOP
   30 DO 35 I=1,IEND-TEMP(3,LAST)
      LIST(1,IEND)=LIST(1,IEND1)
      LIST(2,IEND)=LIST(2,IEND1)
      IEND=IEND-1
   35 IEND1=IEND1-1
      GO TO 25
C     HERE TO OUTPUT COMPLETED PATTERN LIST
   40 CALL SORT(LIST,NOUT,2,2)
      DO 50 I=1,NOUT
      CALL UNPACK(LIST(2,I),IN,NI)
      WRITE(13,82)(IN(J),J=1,NI)
      IF(I.EQ.MAXIN)WRITE(13,80)BLANK(1:NI)
   50 CONTINUE
      WRITE(9,80)'NORMAL TERMINATION'
      PAUSE
      STOP
      END
      INTEGER FUNCTION IPACK(IN,NI)
      INTEGER IN(NI)
      IPACK=0
      DO 5 I=1,NI
    5 IPACK=2*IPACK+IN(I)
      RETURN
      END
      SUBROUTINE UNPACK(J1,IN,NI)
      INTEGER IN(NI)
      COMMON/START/ISTART
      I=1
      J=J1
      IBIT=ISTART
    1 IF(J-IBIT)5,10,10
    5 IN(I)=0
      GO TO 15
```

```
   10 IN(I)=1
      J=J-IBIT
   15 I=I+1
      IBIT=IBIT/2
      IF(IBIT.GT.0)GO TO 1
      RETURN
      END
      SUBROUTINE SEARCH(NEW,LIST,N,IRES,IPT)
C     RETURN 0 IF PATTERN IS IN LIST, ELSE RETURN POINTER TO NEXT HIGHER
      INTEGER LIST(2,N)
      IL=1
      IH=N
    1 I=(IL+IH)/2
      IF(NEW-LIST(2,I))5,10,15
    5 IH=I-1
      IF(IH.GE.IL)GO TO 1
      IRES=1
      IPT=I
      RETURN
   10 IRES=0
      IPT=I
      RETURN
   15 IL=I+1
      IF(IL.LE.IH)GO TO 1
      IRES=1
      IPT=I+1
      RETURN
      END
      INTEGER FUNCTION INTER(IN1X,IN2X)
      COMMON/START/ISTART
      IN1=IN1X
      IN2=IN2X
      IBIT=ISTART
      INTER=0
    1 IF(IN1.LT.IBIT)GO TO 5
      IN1=IN1-IBIT
      IF(IN2.LT.IBIT)GO TO 10
      IN2=IN2-IBIT
      INTER=INTER+IBIT
      GO TO 10
    5 IF(IN2.LT.IBIT)GO TO 10
      IN2=IN2-IBIT
   10 IBIT=IBIT/2
      IF(IBIT.GT.0)GO TO 1
      RETURN
      END
      SUBROUTINE SORT(IBUF,N,LEN,ISKIP)
C     GENERAL-PURPOSE SORTING SUBROUTINE, USING 'QUICK SORT' ALGORITHM
C     IBUF IS VECTOR TO BE SORTED, N IS NUMBER OF ELEMENTS IN VECTOR,
C     LEN IS NUMBER OF WORDS IN EACH ELEMENT, AND ISKIP (≥ LEN) IS NUMBER
C     OF WORDS SEPARATING THE BEGINNING OF SUCCESSIVE VECTOR ELEMENTS
C     LEN MUST BE LESS THAN OR EQUAL TO THE DIMENSION OF PIVOT
      INTEGER IBUF(ISKIP,N),LV(40),UV(40),PIVOT,P
      LV(1)=1 ; UV(1)=N ; P=1
      DO WHILE (P.GT.0)
      IF(LV(P).GE.UV(P)) THEN
        P=P-1
      ELSE
```

BEST COPY AVAILABLE

```
      I=LV(P)-1
      J=UV(P)
      PIVOT=J
    1 I=I+1
      IF(I.GE.J) GO TO 10
      DO 2 KX=1,LEN
      IF(IBUF(KX,I)-IBUF(KX,PIVOT))1,2,4
    2   CONTINUE
        GO TO 1
    4 J=J-1
      IF(J.LE.I) GO TO 10
      DO 6 KX=1,LEN
      IF(IBUF(KX,J)-IBUF(KX,PIVOT))7,6,4
    6   CONTINUE
        GO TO 4
    7 DO 8 KX=1,LEN
          L=IBUF(KX,I)
          IBUF.(KX,I)=IBUF(KX,J)
    8     IBUF(KX,J)=L
      GO TO 1
   10 DO 12 KX=1,LEN
         IF(IBUF(KX,I)-IBUF(KX,PIVOT))13,12,14
   12 CONTINUE
   13 IF(I.EQ.PIVOT)GO TO 16
      I=I+1
   14 DO 15 KX=1,LEN
      L=IBUF(KX,I)
      IBUF(KX,I)=IBUF(KX,PIVOT)
   15 IBUF(KX,PIVOT)=L
   16 IF (I-LV(P) .LT. UV(P)-I)THEN
       LV(P+1)=LV(P)
       UV(P+1)=I-1
       LV(P)=I+1
      ELSE
       LV(P+1)=I+1
       UV(P+1)=UV(P)
       UV(P)=I-1
      END IF
      P=P+1
      END IF
      REPEAT
      RETURN
      END
```

```
C     DISATT
C
C     PROGRAM TO INPUT A LIST OF MANIFEST RESPONSE PATTERNS, WITH
C     FREQUENCIES, AND A SET OF HYPOTHESIZED TP AND FP RATE VECTORS
C     AND CALCULATE LATENT FREQUENCIES FOR ALL POSSIBLE RESPONSE PATTERNS.
C     THESE ARE THEN SORTED FROM MOST FREQUENT DOWN, AND OUTPUT.  OUTPUT
C     FORMAT IS (F12.3,3X,10I1) (PROGRAM IS LIMITED TO TEN ITEMS)
C
C             ************************************************
C             *                                              *
C             *      AUTHOR:  EDWARD H. HAERTEL              *
C             *               STANFORD UNIVERSITY            *
C             *               STANFORD, CA  94305-3096       *
C             *               HAERTEL@LELAND.STANFORD.EDU    *
C             *               415-725-1251                   *
C             *               415-725-7412 (FAX)             *
C             *                                              *
C             *     PERMISSION IS GRANTED TO USE THIS PROGRAM *
C             *     FOR NONCOMMERCIAL PURPOSES PROVIDED THE  *
C             *     ORIGINAL AUTHOR IS CREDITED.  NO WARRANTY *
C             *     AS TO THE ACCURACY OR UTILITY OF THE     *
C             *     PROGRAM FOR ANY PURPOSE IS EXPRESSED     *
C             *     OR IMPLIED.  PERMISSION IS ALSO GRANTED  *
C             *     TO MODIFY THE PROGRAM AND/OR TO INCORPORATE *
C             *     THIS CODE OR ALGORITHMS INTO OTHER       *
C             *     PROGRAMS FOR NONCOMMERCIAL PURPOSES.     *
C             *                                              *
C             ************************************************
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 OMAT(1024),CMATS(10,2,2)
      INTEGER PMAP(1024,10),SORTMAP(2,1024),IN(10)
      CHARACTER*80 FILENAME,FORMAT
      CHARACTER*20 ANSWER
      LOGICAL EXIST
      WRITE(9,80)'NUMBER OF ITEMS'
      READ(9,81)NI
   80 FORMAT(A)
   81 FORMAT(I20)
   82 FORMAT(F20.0)
      NPAT=2**NI
      DO 5 I=1,NPAT
    5 OMAT(I)=0.D0
C     CREATE PMAP
      DO 15 IP=1,NPAT
      J=IP-1
      DO 10 II=1,NI
      I=NI+1-II
      PMAP(IP,I)=J-2*(J/2)+1
   10 J=J/2
   15 CONTINUE
      WRITE(9,80)'NUMBER OF (MANIFEST) PATTERNS'
      WRITE(9,80)'(WITH FREQUENCIES) TO BE INPUT'
      READ(9,81)NIN
   17 WRITE(9,80)'INPUT FILENAME'
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(EXIST)GO TO 20
```

```
      WRITE(9,80)'FILE NOT FOUND'
      GO TO 17
   20 OPEN(11,FILE=FILENAME)
      WRITE(9,80)'INPUT FORMAT:  FREQ (TYPE F) THEN PATTERN VECTOR (TYPE
     * I)'
      READ(9,80)FORMAT
      WRITE(9,80)'OUTPUT FILENAME FOR LATENT FREQUENCY TABLE'
      READ(9,80)FILENAME
      OPEN(12,FILE=FILENAME)
      WRITE(9,80)'ARE TRUE POSITIVE RATES THE SAME FOR ALL ITEMS? (Y/N)'
      READ(9,80)ANSWER
      IF(ANSWER(1:1).NE.'Y'.AND.ANSWER(1:1).NE.'y')GO TO 25
      WRITE(9,80)'COMMON TRUE POSITIVE PROBABILITY'
      READ(9,82)TP
      DO 22 I=1,NI
      CMATS(I,2,2)=TP
   22 CMATS(I,1,2)=1.D0-TP
      GO TO 30
   25 DO 27 I=1,NI
      WRITE(9,84)'TP',I
   84 FORMAT(A2,' FOR ITEM',I2)
      READ(9,82)CMATS(I,2,2)
   27 CMATS(I,1,2)=1.D0-CMATS(I,2,2)
   30 WRITE(9,80)'ARE FALSE POSITIVE RATES THE SAME FOR ALL ITEMS? (Y/N)
     *'
      READ(9,80)ANSWER
      IF(ANSWER(1:1).NE.'Y'.AND.ANSWER(1:1).NE.'y')GO TO 35
      WRITE(9,80)'COMMON FALSE POSITIVE PROBABILITY'
      READ(9,82)FP
      DO 32 I=1,NI
      CMATS(I,2,1)=FP
   32 CMATS(I,1,1)=1.D0-FP
      GO TO 40
   35 DO 37 I=1,NI
      WRITE(9,84)'FP',I
      READ(9,82)CMATS(I,2,1)
   37 CMATS(I,1,1)=1.D0-CMATS(I,2,1)
C     NEXT INVERT ALL 2X2 MATRICES
   40 DO 45 I=1,NI
      DET=CMATS(I,1,1)*CMATS(I,2,2)-CMATS(I,2,1)*CMATS(I,1,2)
      TP=CMATS(I,2,2)
      CMATS(I,2,2)=CMATS(I,1,1)/DET
      CMATS(I,1,1)=TP/DET
      CMATS(I,1,2)=-CMATS(I,1,2)/DET
   45 CMATS(I,2,1)=-CMATS(I,2,1)/DET
C     PROCESS RESPONSE PATTERNS
      DO 60 II=1,NIN
      READ(11,FORMAT)FREQ,(IN(I),I=1,NI)
      DO 52 I=1,NI
   52 IN(I)=IN(I)+1
      DO 55 IP=1,NPAT
      R=FREQ
      DO 50 I=1,NI
   50 R=R*CMATS(I,PMAP(IP,I),IN(I))
   55 OMAT(IP)=OMAT(IP)+R
   60 CONTINUE
C     SET UP AND PERFORM SORT
      WRITE(9,80)'BEGINNING SORT'
```

```
      DO 65 IP=1,NPAT
      SORTMAP(1,IP)=-1000.*OMAT(IP)
   65 SORTMAP(2,IP)=IP
      CALL SORT(SORTMAP,NPAT,2,2)
C     SET UP, THEN OUTPUT
      DO 67 IP=1,NPAT
      DO 67 I=1,NI
   67 PMAP(IP,I)=PMAP(IP,I)-1
      DO 70 II=1,NPAT
      IP=SORTMAP(2,II)
   70 WRITE(12,85)OMAT(IP),(PMAP(IP,I),I=1,NI)
   85 FORMAT(F12.3,3X,10I1)
      WRITE(9,80)'NORMAL TERMINATION.  HIT RETURN TO EXIT.'
      READ(9,80)ANSWER
      STOP
      END
            SUBROUTINE SORT(IBUF,N,LEN,ISKIP)
C     GENERAL-PURPOSE SORTING SUBROUTINE, USING 'QUICK SORT' ALGORITHM
C     IBUF IS VECTOR TO BE SORTED, N IS NUMBER OF ELEMENTS IN VECTOR,
C     LEN IS NUMBER OF WORDS IN EACH ELEMENT, AND ISKIP (≥ LEN) IS NUMBER
C     OF WORDS SEPARATING THE BEGINNING OF SUCCESSIVE VECTOR ELEMENTS
C     LEN MUST BE LESS THAN OR EQUAL TO THE DIMENSION OF PIVOT
      INTEGER IBUF(ISKIP,N),LV(40),UV(40),PIVOT,P
      LV(1)=1 ; UV(1)=N ; P=1
      DO WHILE (P.GT.0)
      IF(LV(P).GE.UV(P)) THEN
        P=P-1
      ELSE
        I=LV(P)-1
        J=UV(P)
        PIVOT=J
    1 I=I+1
      IF(I.GE.J) GO TO 10
      DO 2 KX=1,LEN
      IF(IBUF(KX,I)-IBUF(KX,PIVOT))1,2,4
    2   CONTINUE
        GO TO 1
    4 J=J-1
      IF(J.LE.I) GO TO 10
      DO 6 KX=1,LEN
      IF(IBUF(KX,J)-IBUF(KX,PIVOT))7,6,4
    6   CONTINUE
        GO TO 4
    7 DO 8 KX=1,LEN
          L=IBUF(KX,I)
          IBUF(KX,I)=IBUF(KX,J)
    8     IBUF(KX,J)=L
      GO TO 1
   10 DO 12 KX=1,LEN
         IF(IBUF(KX,I)-IBUF(KX,PIVOT))13,12,14
   12 CONTINUE
   13 IF(I.EQ.PIVOT)GO TO 16
      I=I+1
   14 DO 15 KX=1,LEN
      L=IBUF(KX,I)
      IBUF(KX,I)=IBUF(KX,PIVOT)
   15 IBUF(KX,PIVOT)=L
   16 IF (I-LV(P) .LT. UV(P)-I)THEN
```

```
 LV(P+1)=LV(P)
 UV(P+1)=I-1
 LV(P)=I+1
ELSE
 LV(P+1)=I+1
 UV(P+1)=UV(P)
 UV(P)=I-1
END IF
P=P+1
END IF
REPEAT
RETURN
END
```

```
C     INTERSECT
C
C     PROGRAM TO SHOW INTERSECTIONS OF PATTERNS
C
C            ************************************************
C            *                                              *
C            *      AUTHOR:  EDWARD H. HAERTEL              *
C            *               STANFORD UNIVERSITY            *
C            *               STANFORD, CA  94305-3096       *
C            *               HAERTEL@LELAND.STANFORD.EDU    *
C            *               415-725-1251                   *
C            *               415-725-7412 (FAX)             *
C            *                                              *
C            *     PERMISSION IS GRANTED TO USE THIS PROGRAM *
C            *     FOR NONCOMMERCIAL PURPOSES PROVIDED THE  *
C            *     ORIGINAL AUTHOR IS CREDITED.  NO WARRANTY *
C            *     AS TO THE ACCURACY OR UTILITY OF THE     *
C            *     PROGRAM FOR ANY PURPOSE IS EXPRESSED     *
C            *     OR IMPLIED.  PERMISSION IS ALSO GRANTED  *
C            *     TO MODIFY THE PROGRAM AND/OR TO INCORPORATE *
C            *     THIS CODE OR ALGORITHMS INTO OTHER       *
C            *     PROGRAMS FOR NONCOMMERCIAL PURPOSES.     *
C            *                                              *
C            ************************************************
C
      IMPLICIT INTEGER(A-Z)
      INTEGER INPAT(2000),IN(30),OUT(2000)
      CHARACTER*80 FILENAME,FORMAT
      LOGICAL EXIST
      COMMON/START/ISTART
   80 FORMAT(A)
      WRITE(9,80)'NUMBER OF ITEMS'
      READ(9,81)NI
      ISTART=2**(NI-1)
    1 WRITE(9,80)'INPUT FILENAME'
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(EXIST)GO TO 5
      WRITE(9,80)FILENAME//' NOT FOUND'
      GO TO 1
    5 OPEN(11,FILE=FILENAME)
      WRITE(9,80)'NUMBER OF PATTERNS TO INPUT'
      READ(9,81)NPAT
   81 FORMAT(I20)
      WRITE(9,80)'INPUT FORMAT (TYPE I)'
      READ(9,80)FORMAT
      DO 10 I=1,NPAT
      READ(11,FORMAT)(IN(J),J=1,NI)
   10 INPAT(I)=IPACK(IN,NI)
   11 WRITE(9,80)'OUTPUT FILE'
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(.NOT.EXIST)GO TO 12
      WRITE(9,80)FILENAME//' ALREADY EXISTS.  PLEASE CHOOSE ANOTHER NAME
     *'
      GO TO 11
   12 OPEN(12,FILE=FILENAME)
      NCHECK=0
```

BEST COPY AVAILABLE

```
 13 WRITE(9,82)NCHECK
 82 FORMAT('HAVE CHECKED',I5,'.  HOW MANY MORE?  (0 TO EXIT)')
    READ(9,81)INCREMENT
    IF(INCREMENT.EQ.0)STOP
    LOW=NCHECK+1
    HIGH=NCHECK+INCREMENT
    NCHECK=HIGH
    DO 20 IBASE=LOW,HIGH
    DO 15 ITEST=1,IBASE
    IPAT=INTER(INPAT(ITEST),INPAT(IBASE))
    DO 14 J=1,NPAT
    IF(IPAT.EQ.INPAT(J))GO TO 15
 14 CONTINUE
    J=9999
 15 OUT(ITEST)=J
 20 WRITE(12,83)IBASE,(OUT(I),I=1,IBASE)
 83 FORMAT(/I5,(T6,30I5))
    GO TO 13
    END
    INTEGER FUNCTION IPACK(IN,NI)
    INTEGER IN(NI)
    IPACK=0
    DO 5 I=1,NI
  5 IPACK=2*IPACK+IN(I)
    RETURN
    END
    INTEGER FUNCTION INTER(IN1X,IN2X)
    COMMON/START/ISTART
    IN1=IN1X
    IN2=IN2X
    IBIT=ISTART
    INTER=0
  1 IF(IN1.LT.IBIT)GO TO 5
    IN1=IN1-IBIT
    IF(IN2.LT.IBIT)GO TO 10
    IN2=IN2-IBIT
    INTER=INTER+IBIT
    GO TO 10
  5 IF(IN2.LT.IBIT)GO TO 10
    IN2=IN2-IBIT
 10 IBIT=IBIT/2
    IF(IBIT.GT.0)GO TO 1
    RETURN
    END
```

```
C     SUPERCLOSE
C
C     'SUPER'-CLOSURE:  PROGRAM FOR INTERACTIVE MODELING OF
C     LATENT RESPONSE PATTERN LATTICE STRUCTURES.
C
C     ******************************************************
C     *                                                    *
C     *       AUTHOR:  EDWARD H. HAERTEL                   *
C     *                STANFORD UNIVERSITY                 *
C     *                STANFORD, CA  94305-3096            *
C     *                HAERTEL@LELAND.STANFORD.EDU         *
C     *                415-725-1251                        *
C     *                415-725-7412 (FAX)                  *
C     *                                                    *
C     *      PERMISSION IS GRANTED TO USE THIS PROGRAM     *
C     *      FOR NONCOMMERCIAL PURPOSES PROVIDED THE       *
C     *      ORIGINAL AUTHOR IS CREDITED.  NO WARRANTY     *
C     *      AS TO THE ACCURACY OR UTILITY OF THE          *
C     *      PROGRAM FOR ANY PURPOSE IS EXPRESSED          *
C     *      OR IMPLIED.  PERMISSION IS ALSO GRANTED       *
C     *      TO MODIFY THE PROGRAM AND/OR TO INCORPORATE   *
C     *      THIS CODE OR ALGORITHMS INTO OTHER            *
C     *      PROGRAMS FOR NONCOMMERCIAL PURPOSES.          *
C     *                                                    *
C     ******************************************************
C
C     THE BASIC IDEA OF THIS PROGRAM IS THAT A SET OF (TRUE OR
C     LATENT) PERFORMANCE STATE PATTERNS REPRESENTS THE UNION OF
C     ONE OR MORE DISTRIBUTIVE LATTICES REPRESENTING UNIQUE ABILITY
C     CONFIGURATIONS ENABLING EXECUTION OF THE RESPECTIVE ITEMS.
C     IF EACH ITEM CAN BE SOLVED USING EXACTLY ONE SET OF UNDERLYING
C     ABILITIES, THEN A SINGLE 'ABILITY STRUCTURE' SHOULD SUFFICE.  IF
C     ONE OR MORE ITEMS CAN BE SOLVED USING ONE OR MORE DISTINCT SKILLS,
C     THEN TWO OR MORE ABILITY STRUCTURES MAY BE REQUIRED, ACCOUNTING
C     FOR (OVERLAPPING) SUBSETS OF THE LATENT ABILITY PATTERNS.
C
C     THIS INTERACTIVE PROGRAM INCORPORATES CODE FROM THE EARLIER
C     (NONINTERACTIVE) 'CLOSURE' PROGRAM, HENCE 'SUPER-CLOSURE'.
C     IT CAN MAINTAIN UP TO 20 DATA STRUCTURES SIMULTANEOUSLY,
C     AND CAN HANDLE LATENT RESPONSE PATTERNS ON UP TO 30 ITEMS.  UP
C     TO 2000 LATENT RESPONSE PATTERNS (WITH ASSOCIATED FREQUENCIES)
C     MAY BE INPUT.
C
C     INPUT IS A FILE CONTAINING A LIST OF 1-0 VECTORS AND
C     ASSOCIATED FREQUENCIES.  THE LIST IS TREATED AS ORDERED;
C     USUALLY ORDER WILL BE DESCENDING FREQUENCY.  THE LIST
C     COULD BE EITHER OF MANIFEST RESPONSE PATTERNS OR OF
C     ESTIMATED FREQUENCIES OF LATENT RESPONSE PATTERNS.  LATTER
C     IS THE USE I HAVE IN MIND AT THE PRESENT.  SUCH A LIST IS
C     GENERATED, FOR EXAMPLE, BY 'DISATT' PROGRAM, WHICH ACCEPTS
C     VECTORS OF TRUE POSITIVE AND FALSE POSITIVE PROBABILITIES
C     FOR A SET OF ITEMS, TOGETHER WITH MANIFEST FREQUENCIES, AND
C     CALCULATES CORRESPONDING LATENT RESPONSE PATTERN FREQUENCIES.
C
C     THE USER OF THIS PROGRAM REQUESTS PATTERNS SEQUENTIALLY FROM
C     THE INPUT LIST, ALTHOUGH THERE IS AN OPTION TO MOVE AROUND
C     IN THE LIST.  PATTERNS ARE REFERRED TO ONLY BY THEIR INPUT
C     LIST SEQUENCE NUMBERS.  IT WILL BE HELPFUL TO HAVE A PRINTOUT
```

```
C     OF THE INPUT LIST AT HAND WHEN RUNNING THIS PROGRAM.
C
C     AFTER REQUESTING NUMBER OF ITEMS, INPUT FILE NAME, AND INPUT
C     FORMAT, PROGRAM READS ENTIRE LIST INTO MEMORY AND REQUESTS A
C     COMMAND.  EACH COMMAND CONSISTS OF A LETTER (UPPER OR LOWER
C     CASE) READ IN COLUMN 1, USUALLY FOLLOWED BY A NUMBER.  ONE OR
C     MORE SPACES BETWEEN THE LETTER AND THE NUMBER ARE OPTIONAL.
C     THE MEANING OF THE NUMBER (IF ANY) DEPENDS ON THE PRECEDING
C     LETTER:
C
C     N     GET NEXT PATTERN.  THIS WILL ALWAYS BE THE FIRST COMMAND,
C           AND RESULTS IN DISPLAY OF "CURRENT PATTERN IS" LINE.
C           AN ALTERNATIVE FORM OF THIS COMMAND IS N<#>.  THIS GETS
C           THE PATTERN AT LOCATION <#> IN THE INPUT LIST.  SUBSEQUENT
C           'N' REQUESTS WILL CONTINUE FROM THERE.  IT IS PERMISSIBLE
C           TO GO BACKWARD OR FORWARD.
C
C     C<#>  CHECK RESULT OF ADDING CURRENT PATTERN TO DATA STRUCTURE
C           INDEXED BY <#>.  ANSWER IS EITHER 'ALREADY IMPLIED' OR ELSE
C           A LIST OF NEW PATTERNS IMPLIED.  FOR EACH OF THESE NEW
C           PATTERNS, IF IT IS IN THE ORIGINAL INPUT LIST, ITS SEQUENCE
C           NUMBER AND ASSOCIATED FREQUENCY ARE GIVEN.
C
C     A<#>  ADD CURRENT PATTERN TO DATA STRUCTURE INDEXED BY <#>.
C           OUTPUT OF THIS COMMAND IS IDENTICAL TO THAT OF C<#>.  IF
C           IMMEDIATELY PRECEDING COMMAND WAS C<#>, THE COMMAND 'A'
C           MAY BE GIVEN WITHOUT A NUMBER.  IN THIS CASE, THE <#> FROM
C           THE 'C' COMMAND IS IMPLIED, AND REDUNDANT OUTPUT IS SUPPRESSED.
C
C     R<#>  ERASES DATA STRUCTURE INDEXED BY <#>.  IF <#> IS OMITTED, ALL
C           DATA STRUCTURES ARE PURGED.
C
C     D<#>  DUPLICATES DATA STRUCTURE INDEXED BY <#>.  THE COPY CREATED
C           WILL BE ASSIGNED THE FIRST AVAILABLE INDEX NUMBER.
C
C     S     TRIGGERS REQUEST FOR OUTPUT FILE NAME, THEN UPDATES,
C           SORTS, AND OUTPUTS ALL CURRENT DATA STRUCTURES.  UPDATING
C           REFERS TO A COMPARISON OF EACH PATTERN LIST TO THE INPUT
C           DATA LIST FROM THE FIRST THROUGH THE CURRENT PATTERN.  FOR
C           EACH MATCH, THE INTERNAL SEQUENCE NUMBER IS UPDATED SO THAT
C           WHEN DATA ARE OUTPUT, EXTANT PATTERNS ARE NOT DENOTED AS
C           GENERATED PATTERNS.  NOTE THAT WORK CAN BE SAVED AT WILL,
C           AT INTERMEDIATE POINTS IN ANALYSIS.  S<#> SAVES ONLY DATA
C           STRUCTURE INDICATED BY <#>.  AT TIME OF SAVE, OPTIONAL
C           COMMENTS MAY BE ENTERED.  THESE ARE SAVED WITH THE DATA
C           STRUCTURE(S).  NOTE THAT UPDATING CAN BE SUPPRESSED BY
C           PRECEDING S COMMAND WITH "N 1"
C
C     O     OPENS NEW FILE FOR DATA OUTPUT.  ONLY ONE FILE CAN BE OPEN
C           AT A TIME, AND ONCE CLOSED, A FILE CANNOT BE REOPENED.
C
C     Q     QUITS IMMEDIATELY WITHOUT SAVING
C
C     E     'EXIT', EQUIVALENT TO S FOLLOWED BY Q
C
C     U     UNDO LAST COMMAND (CANNOT UNDO 'S', 'O', 'Q', 'E', OR 'U' ITSELF;
C           UNDO IS MEANINGLESS FOR 'C'.)
C
```

```
C     L    LISTS ALL PATTERNS INCLUDED IN A GIVEN STRUCTURE
C
C     M    GENERATES LATTICE MAP OF A GIVEN STRUCTURE
C
C     I    LISTS INDEX NUMBERS OF ALL PATTERNS IN STRUCTURE
C
      IMPLICIT INTEGER(A-Z)
      INTEGER LIST(2,2000,20),TEMP(3,1000),IN(30),INPAT(2000),
     * LENGTH(20),NUMREQ(13),LENGTHOLD(20),BACKUPLIST(2,2000),
     * LBUFFER(30,27),NDEF(13)
      REAL*8 FREQ(2000)
      CHARACTER*80 FILENAME,FORMAT
      CHARACTER*1 CMD,CODES(26)
      LOGICAL EXIST,OUTFILE,DISPLAY,SAVED(20),SAVEDHOLD,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
     * OUTFILE,LOG
      DATA CODES/'A','a','C','c','D','d','E','e','N','n','O','o',
     * 'Q','q','R','r','S','s','U','u','L','l','M','m','I','i'/,
     * LENGTH/20*0/
     * NUMREQ/0,0,1,10*0/,NDEF/1,1,8*0,3*1/
C     (NUMREQ(I)=1 IFF COMMAND I REQUIRES A NUMBER AS AN ARGUMENT.)
C     (NDEF(I)=1 IFF COMMAND I REQUIRES NUMERIC ARGUMENT BUT MAY DEFAULT)
      OUTFILE=.FALSE.
      WRITE(9,80)'ENTER FILENAME FOR SESSION LOG (OR JUST <CR> IF NO LOG
     * DESIRED)'
      READ(9,80)FILENAME
      LOG=LEN(TRIM(FILENAME)).GT.0
      IF(.NOT.LOG)GO TO 4
   40 INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(.NOT.EXIST)GO TO 41
      WRITE(9,80)TRIM(FILENAME)//' ALREADY EXISTS'
      WRITE(9,80)'PLEASE CHOOSE ANOTHER NAME'
      READ(9,80)FILENAME
      GO TO 40
   41 OPEN (1,FILE=FILENAME)
    4 WRITE(9,80)'NUMBER OF ITEMS IN EACH PATTERN'
      IF(LOG)WRITE(1,80)'NUMBER OF ITEMS IN EACH PATTERN'
      READ(9,81)NI
      IF(LOG)WRITE(1,811)NI
  811 FORMAT(I2)
   80 FORMAT(A)
   81 FORMAT(I20)
      ISTART=2**(NI-1)
      WRITE(9,80)'NUMBER OF PATTERNS TO BE READ'
      IF(LOG)WRITE(1,80)'NUMBER OF PATTERNS TO BE READ'
      READ(9,81)MAXIN
      IF(LOG)WRITE(1,812)MAXIN
  812 FORMAT(I4)
      ISER=MAXIN+1
    2 WRITE(9,80)'INPUT FILENAME'
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(EXIST)GO TO 3
      WRITE(9,80)'FILE NOT FOUND'
      GO TO 2
    3 OPEN(11,FILE=FILENAME)
      IF(LOG)WRITE(1,80)'INPUT FILENAME'
      IF(LOG)WRITE(1,80)FILENAME
```

```
      WRITE(9,80)'INPUT DATA FORMAT SHOULD DESCRIBE PATTERN VECTOR'
      WRITE(9,80)'(I FORMAT) FOLLOWED BY FREQUENCY (F FORMAT).'
      WRITE(9,80)'INPUT FORMAT'
      READ(9,80)FORMAT
      IF(LOG)WRITE(1,80)'INPUT FORMAT'
      IF(LOG)WRITE(1,80)FORMAT
      DO 5 I=1,MAXIN
      READ(11,FORMAT)(IN(J),J=1,NI),FREQ(I)
    5 INPAT(I)=IPACK(IN,NI)
      WRITE(9,80)'INPUT FILE LOADED'
      WRITE(9,80)' '
      IF(LOG)WRITE(1,80)'INPUT FILE LOADED'
      IF(LOG)WRITE(1,80)' '
      ICOM=0
      NEXT=0
      NUM=0
C     HERE AT BEGINNING OF COMMAND LOOP
    1 WRITE(9,80)'>'
      READ(9,82)CMD,NUMT
   82 FORMAT(A1,I20)
      IF(LOG)WRITE(1,821)CMD,NUMT
  821 FORMAT(A1,I5)
      DO 10 I=1,26
      IF(CMD.EQ.CODES(I))GO TO 15
   10 CONTINUE
      WRITE(9,80)CMD//' NOT RECOGNIZED AS VALID COMMAND'
      IF(LOG)WRITE(1,80)CMD//' NOT RECOGNIZED AS VALID COMMAND'
      GO TO 1
   15 CTEMP=(I+1)/2
      IF(NUMT.GT.0.OR.NUMREQ(CTEMP).EQ.0)GO TO 20
      WRITE(9,80)CMD//' MUST BE FOLLOWED BY A NUMBER'
      IF(LOG)WRITE(1,80)CMD//' MUST BE FOLLOWED BY A NUMBER'
      GO TO 1
C     CHECK FOR 'UNDO', ARGUMENT IN RANGE, ETC.
   20 IF(CTEMP.NE.10)GO TO 30
      IF(ICOM.LT.1)GO TO 29
      GO TO (21,29,23,29,25,29,29,28,29,29,29,29,29),ICOM
C     UNDO 'A'
   21 LENGTH(NUM)=LENHOLD
      SAVED(NUM)=SAVEDHOLD
      DO 211 I=1,2
      DO 211 J=1,LENGTH(NUM)
  211 LIST(I,J,NUM)=BACKUPLIST(I,J)
      WRITE(9,80)'A COMMAND UNDONE'
      IF(LOG)WRITE(1,80)'A COMMAND UNDONE'
      ICOM=CTEMP
      GO TO 1
C     UNDO 'D'
   23 IF(NEWHOLD.EQ.0)GO TO 231
      LENGTH(NEWHOLD)=0
  231 WRITE(9,80)'D COMMAND UNDONE'
      IF(LOG)WRITE(1,80)'D COMMAND UNDONE'
      ICOM=CTEMP
      GO TO 1
C     UNDO 'N'
   25 NEXT=NEXTHOLD
      WRITE(9,80)'N COMMAND UNDONE'
      IF(LOG)WRITE(1,80)'N COMMAND UNDONE'
```


BEST COPY AVAILABLE

```
      ICOM=CTEMP
      GO TO 1
C     UNDO 'R'
   28 IF(NUM.GT.0)GO TO 285
C     HERE TO UNDO PURGE
      DO 281 I=1,20
  281 LENGTH(I)=LENGTHOLD(I)
      GO TO 288
  285 LENGTH(NUM)=LENHOLD
  288 WRITE(9,80)'R COMMAND UNDONE'
      IF(LOG)WRITE(1,80)'R COMMAND UNDONE'
      ICOM=CTEMP
      GO TO 1
   29 WRITE(9,80)'CAN''T UNDO'
      IF(LOG)WRITE(1,80)'CAN''T UNDO'
      ICOM=CTEMP
      GO TO 1
   30 IF(NUMT.GT.MAXIN)GO TO 35
C     IF INITIAL COMMAND IS NOT 'N', ISSUE DEFAULT N
      IF(CTEMP.EQ.5)GO TO 31
      IF(NEXT.EQ.0)NEXT=1
C     FINISH CHECKING FOR ARGUMENT IN RANGE
      IF(NUMT.GT.20)GO TO 35
      IF(NUMT.EQ.0)GO TO 31
      IF(LENGTH(NUMT).GT.0)GO TO 31
      IF(CTEMP.EQ.1)GO TO 31
      WRITE(9,80)'FIRST REFERENCE TO NEW DATA STRUCTURE MUST BE ''A'' CO
     *MMAND'
      IF(LOG)WRITE(1,80)'FIRST REFERENCE TO NEW DATA STRUCTURE MUST BE '
     * //'''A'' COMMAND'
      GO TO 1
   35 WRITE(9,80)'NUMERIC VALUE OUT OF RANGE'
      IF(LOG)WRITE(1,80)'NUMERIC VALUE OUT OF RANGE'
      GO TO 1
C     CHECK FOR 'DEFAULT' DATA STRUCTURE REFERENCE
   31 IF(NDEF(CTEMP).EQ.0)GO TO 33
      IF(NUMT.GT.0)GO TO 33
      IF(NUM.EQ.0.OR.NUM.GT.20)GO TO 34
      IF(LENGTH(NUM).GT.0)GO TO 32
   34 WRITE(9,80)'UNABLE TO FIND DEFAULT DATA STRUCTURE REFERENCE FOR '
     * //CMD//' COMMAND'
      IF(LOG)WRITE(1,80)'UNABLE TO FIND DEFAULT DATA STRUCTURE '//
     * 'REFERENCE FOR '//CMD//' COMMAND'
      GO TO 1
   32 NUMT=NUM
C     ACCEPT COMMAND
   33 ICOMP=ICOM
      ICOM=CTEMP
      NUMP=NUM
      NUM=NUMT
      GO TO (1000,2000,3000,4000,5000,6000,7000,8000,9000,7000,11000,
     * 12000,13000),ICOM
C
C     HERE FOR 'A' (ADD) COMMAND
C
 1000 LENHOLD=LENGTH(NUM)
      DO 1001 I=1,2
      DO 1001 J=1,LENHOLD
```

```
 1001 BACKUPLIST(I,J)=LIST(I,J,NUM)
C     SUPPRESS DISPLAY IF IMMEDIATELY PRECEDING COMMAND WAS C<NUM>
      DISPLAY=NUMP.NE.NUM.OR.ICOMP.NE.2
      CALL CLOSURE(LIST(1,1,NUM),LENGTH(NUM),NEXT,DISPLAY,.TRUE.)
      IF(.NOT.DISPLAY)WRITE(9,80)'PATTERN ADDED'
      IF(.NOT.DISPLAY.AND.LOG)WRITE(1,80)'PATTERN ADDED'
      SAVEDHOLD=SAVED(NUM)
      SAVED(NUM)=.FALSE.
      GO TO 1
C
C     HERE FOR 'C' (CHECK) COMMAND
C
 2000 CALL CLOSURE(LIST(1,1,NUM),LENGTH(NUM),NEXT,.TRUE.,.FALSE.)
      GO TO 1
C
C     HERE FOR 'D' (DUPLICATE) COMMAND
C
 3000 DO 3001 I=1,20
      IF(LENGTH(I).EQ.0)GO TO 3002
 3001 CONTINUE
      WRITE(9,80)'D COMMAND FAILED--NO MORE SLOTS AVAILABLE'
      IF(LOG)WRITE(1,80)'D COMMAND FAILED--NO MORE SLOTS AVAILABLE'
      NEWHOLD=0
      GO TO 1
 3002 NEWHOLD=I
      LENGTH(NEWHOLD)=LENGTH(NUM)
      DO 3005 I=1,2
      DO 3005 J=1,LENGTH(NUM)
 3005 LIST(I,J,NEWHOLD)=LIST(I,J,NUM)
      WRITE(9,3080)NUM,NEWHOLD
      IF(LOG)WRITE(1,3080)NUM,NEWHOLD
 3080 FORMAT('DATA STRUCTURE',I3,' COPIED TO SLOT',I3)
      SAVED(NEWHOLD)=SAVED(NUM)
      GO TO 1
C
C     HERE FOR 'E' (EXIT) COMMAND
C
 4000 DO 4001 I=1,20
      IF(LENGTH(I).GT.0.AND..NOT.SAVED(I))CALL SAVE(LIST(1,1,I),
     * LENGTH(I),I)
 4001 CONTINUE
      WRITE(9,80)'HIT <CR> FOR EXIT'
      IF(LOG)WRITE(1,80)'HIT <CR> FOR EXIT'
      READ(9,80)FILENAME
      STOP
C
C     HERE FOR 'N' (NEXT) COMMAND
C
 5000 NEXTHOLD=NEXT
      IF(NUM.EQ.0)NEXT=NEXT+1
      IF(NUM.GT.0)NEXT=NUM
      IF(NUM.EQ.0)NUM=NUMP
      CALL UNPACK(INPAT(NEXT),IN,NI)
      WRITE(9,5080)NEXT,FREQ(NEXT),(IN(I),I=1,NI)
      IF(LOG)WRITE(1,5080)NEXT,FREQ(NEXT),(IN(I),I=1,NI)
 5080 FORMAT('PATTERN',I5,' (',F10.4,') = ',30I1)
      GO TO 1
C
```

```
C     HERE FOR 'O' (OPEN) COMMAND
C
 6000 CALL OPENOUT
      GO TO 1
C
C     HERE FOR Q (QUIT) COMMAND
C
 7000 WRITE(9,80)'HIT <CR> FOR EXIT'
      IF(LOG)WRITE(1,80)'HIT <CR> FOR EXIT'
      READ(9,80)FILENAME
      STOP
C
C     HERE FOR 'R' (REMOVE) COMMAND
C
 8000 IF(NUM.GT.0)GO TO 8010
      DO 8005 I=1,20
      LENGTHOLD(I)=LENGTH(I)
 8005 LENGTH(I)=0
      WRITE(9,80)'ALL DATA STRUCTURES PURGED'
      IF(LOG)WRITE(1,80)'ALL DATA STRUCTURES PURGED'
      GO TO 1
 8010 LENHOLD=LENGTH(NUM)
      LENGTH(NUM)=0
      WRITE(9,8080)NUM
      IF(LOG)WRITE(1,8080)NUM
 8080 FORMAT('DATA STRUCTURE',I3,' PURGED')
      GO TO 1
C
C     HERE FOR 'S' (SAVE) COMMAND
C
 9000 IF(NUM.EQ.0)GO TO 9010
      CALL SAVE(LIST(1,1,NUM),LENGTH(NUM),NUM)
      SAVED(NUM)=.TRUE.
      GO TO 1
 9010 DO 9020 I=1,20
      IF(LENGTH(I).GT.0.AND..NOT.SAVED(I))CALL SAVE(LIST(1,1,I),
     * LENGTH(I),I)
      SAVED(I)=.TRUE.
 9020 CONTINUE
      GO TO 1
C
C     HERE FOR 'L' (LIST) COMMAND
C
11000 NREP=85/(NI+7)
      ENCODE(18,11080,FORMAT)NREP,NI
11080 FORMAT('((',I2,'(I5,2X,',I2,'I1)))')
      LINES=(LENGTH(NUM)+NREP-1)/NREP
      CALL UPDATE(LIST(1,1,NUM),LENGTH(NUM))
      DO 11100 ILINE=1,LINES
      LLIM=1+NREP*(ILINE-1)
      ULIM=LLIM+NREP-1
      IF(ULIM.GT.LENGTH(NUM))ULIM=LENGTH(NUM)
      J=0
      DO 11050 I=LLIM,ULIM
      J=J+1
11050 CALL UNPACK(LIST(2,I,NUM),LBUFFER(1,J),NI)
      WRITE(9,FORMAT)(LIST(1,L+LLIM-1,NUM),(LBUFFER(I,L),I=1,NI),L=1,J)
      IF(LOG)
```

```
     *WRITE(1,FORMAT)(LIST(1,L+LLIM-1,NUM),(LBUFFER(I,L),I=1,NI),L=1,J)
11100 CONTINUE
      GO TO 1
C
C     HERE FOR 'M' (MAP) COMMAND
C
12000 CALL UPDATE(LIST(1,1,NUM),LENGTH(NUM))
      CALL MAP(LIST(1,1,NUM),LENGTH(NUM),NI)
      GO TO 1
C
C     HERE FOR 'I' (INDEX) COMMAND
C
13000 CALL UPDATE(LIST(1,1,NUM),LENGTH(NUM))
      DO 13005 I=1,LENGTH(NUM)
13005 BACKUPLIST(1,I)=LIST(1,I,NUM)
      CALL SORT(BACKUPLIST,LENGTH(NUM),1,2)
      WRITE(9,13080)(BACKUPLIST(1,I),I=1,LENGTH(NUM))
      IF(LOG)WRITE(1,13080)(BACKUPLIST(1,I),I=1,LENGTH(NUM))
13080 FORMAT(15I5)
      GO TO 1
      END
      SUBROUTINE CLOSURE(LIST,NIN,NEXT,DISPLAY,UPDATE)
      REAL*8 FREQ(2000)
      INTEGER LIST(2,5000),TEMP(3,1000),INPAT(2000),BACKUPLIST(2,2000),
     * IN(30)
      CHARACTER*80 FILENAME
      LOGICAL OUTFILE,EXIST,DISPLAY,UPDATE,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
     * OUTFILE,LOG
   80 FORMAT(A)
      IF(NIN.GT.0)GO TO 9
C     FIRST CASE
      IF(.NOT.DISPLAY)GO TO 3
      CALL UNPACK(INPAT(NEXT),IN,NI)
      WRITE(9,81)0,1
      IF(LOG)WRITE(1,81)0,1
   81 FORMAT('FROM',I4,' PATTERNS TO',I4)
      WRITE(9,80)'NEW PATTERNS:'
      IF(LOG)WRITE(1,80)'NEW PATTERNS:'
      WRITE(9,82)NEXT,(IN(I),I=1,NI)
      IF(LOG)WRITE(1,82)NEXT,(IN(I),I=1,NI)
    3 IF(.NOT.UPDATE)RETURN
      LIST(1,1)=NEXT
      LIST(2,1)=INPAT(NEXT)
      NIN=1
      RETURN
   82 FORMAT(I4,2X,30I1)
C     CHECK IF INPAT(NEXT) IS ALREADY IN THE LIST, AND STORE IN TEMP IF NOT
    9 CALL SEARCH(INPAT(NEXT),LIST,NIN,IRES,IPT)
      IF(IRES.GT.0)GO TO 5
C     NEW PATTERN HAS ALREADY BEEN ADDED--REPLACE SERIAL NUMBER
      LIST(1,IPT)=NEXT
      IF(DISPLAY)WRITE(9,80)'PATTERN ALREADY IN LIST'
      IF(DISPLAY.AND.LOG)WRITE(1,80)'PATTERN ALREADY IN LIST'
      RETURN
    5 TEMP(1,1)=INPAT(NEXT)
      TEMP(2,1)=NEXT
      TEMP(3,1)=IPT
```

BEST COPY AVAILABLE

```
      LAST=1
C     CHECK INTERSECTIONS WITH ALL PATTERNS ALREADY IN LIST
      DO 10 I=1,NIN
      J=INTER(INPAT(NEXT),LIST(2,I))
      CALL SEARCH(J,LIST,NIN,IRES,IPT)
      IF(IRES.EQ.0)GO TO 10
C     HERE TO ADD NEW PATTERN TO LIST
      LAST=LAST+1
      TEMP(1,LAST)=J
      TEMP(2,LAST)=ISER
      TEMP(3,LAST)=IPT
      ISER=ISER+1
   10 CONTINUE
C     SORT ADDITIONAL PATTERNS, ELIMINATE DUPLICATES, AND UPDATE LIST
      KEEP=1
      IF(LAST.EQ.1)GO TO 16
      CALL SORT(TEMP,LAST,3,3)
      DO 15 I=2,LAST
      IF(TEMP(1,I-1).NE.TEMP(1,I))GO TO 12
      TEMP(3,I)=-1
      GO TO 15
   12 TEMP(3,I)=TEMP(3,I)+KEEP
      KEEP=KEEP+1
   15 CONTINUE
   16 IF(.NOT.DISPLAY)GO TO 161
      WRITE(9,81)NIN,NIN+KEEP
      IF(LOG)WRITE(1,81)NIN,NIN+KEEP
      WRITE(9,80)'NEW PATTERNS:'
      IF(LOG)WRITE(1,80)'NEW PATTERNS:'
      DO 163 I=1,LAST
      IF(TEMP(3,I).LT.0)GO TO 163
      CALL UNPACK(TEMP(1,I),IN,NI)
C     GET INDEX NUMBER OF NEW PATTERNS (IF ANY)
      DO 170 INDEX=1,MAXIN
      IF(TEMP(1,I).EQ.INPAT(INDEX))GO TO 171
  170 CONTINUE
      INDEX=0
  171 WRITE(9,82)INDEX,(IN(J),J=1,NI)
      IF(LOG)WRITE(1,82)INDEX,(IN(J),J=1,NI)
  163 CONTINUE
  161 IF(.NOT.UPDATE)RETURN
      IEND1=NIN
      NIN=NIN+KEEP
      IEND=NIN
   17 IF(TEMP(3,LAST).GT.0)GO TO 18
      LAST=LAST-1
      GO TO 17
   18 IF(IEND-TEMP(3,LAST))20,25,30
   20 WRITE(9,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION ABORTED (1)'
      IF(LOG)WRITE(1,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION AB'//
     *'ORTED (1)'
      PAUSE
      STOP
   25 LIST(1,IEND)=TEMP(2,LAST)
      LIST(2,IEND)=TEMP(1,LAST)
      IEND=IEND-1
      LAST=LAST-1
      IF(LAST.GT.0)GO TO 17
```

```
      IF(IEND.EQ.IEND1)RETURN
      WRITE(9,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION ABORTED (2)'
      IF(LOG)WRITE(1,80)'A SERIOUS ERROR HAS OCCURRED.  EXECUTION AB'//
     *'ORTED (2)'
      PAUSE
      STOP
   30 DO 35 I=1,IEND-TEMP(3,LAST)
      LIST(1,IEND)=LIST(1,IEND1)
      LIST(2,IEND)=LIST(2,IEND1)
      IEND=IEND-1
   35 IEND1=IEND1-1
      GO TO 25
      END
      SUBROUTINE SAVE(LIST,LENGTH,ID)
      REAL*8 FREQ(2000)
      INTEGER INPAT(2000),LIST(2,2000),BACKUPLIST(2,2000),IN(30)
      CHARACTER*80 COMMENT
      LOGICAL OUTFILE,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
     * OUTFILE,LOG
      IF(.NOT.OUTFILE)CALL OPENOUT
      WRITE(9,81)ID
      IF(LOG)WRITE(1,81)ID
   81 FORMAT('SAVING DATA STRUCTURE',I3)
      WRITE(12,82)ID
   82 FORMAT('***** S T R U C T U R E   ',I2,' *****')
      WRITE(9,80)'ENTER (OPTIONAL) COMMENTS; TERMINATE WITH <CR>'
      IF(LOG)WRITE(1,80)'ENTER (OPTIONAL) COMMENTS; TERMINATE '//
     *'WITH <CR>'
    1 WRITE(9,80)'?  '
      READ(9,80)COMMENT
      IF(LEN(TRIM(COMMENT)).EQ.0)GO TO 5
      WRITE(12,80)COMMENT
      IF(LOG)WRITE(1,80)COMMENT
      GO TO 1
   80 FORMAT(A)
    5 DO 10 I=1,LENGTH
      IF(LIST(1,I).LE.MAXIN)GO TO 9
C     SEARCH TO SEE IF PATTERN OCCURRED IN INPUT LIST
      DO 6 J=1,MAXIN
      IF(LIST(2,I).EQ.INPAT(J))GO TO 7
    6 CONTINUE
      GO TO 9
    7 LIST(1,I)=J
    9 BACKUPLIST(1,I)=LIST(1,I)
   10 BACKUPLIST(2,I)=LIST(2,I)
      CALL SORT(BACKUPLIST,LENGTH,2,2)
      DO 15 I=1,LENGTH
      CALL UNPACK(BACKUPLIST(2,I),IN,NI)
   15 WRITE(12,83)BACKUPLIST(1,I),(IN(J),J=1,NI)
   83 FORMAT(I5,1X,30I1)
      RETURN
      END
      SUBROUTINE UPDATE(LIST,LENGTH)
      REAL*8 FREQ(2000)
      INTEGER INPAT(2000),LIST(2,2000),BACKUPLIST(2,2000)
      LOGICAL OUTFILE,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
```

```
     * OUTFILE,LOG
      DO 11010 I=1,LENGTH
      IF(LIST(1,I).LE.MAXIN)GO TO 11010
      DO 11005 J=1,MAXIN
      IF(LIST(2,I).EQ.INPAT(J))GO TO 11006
11005 CONTINUE
      GO TO 11010
11006 LIST(1,I)=J
11010 CONTINUE
      RETURN
      END
      SUBROUTINE OPENOUT
      REAL*8 FREQ(2000)
      INTEGER INPAT(2000),BACKUPLIST(2,2000)
      CHARACTER*80 FILENAME
      LOGICAL OUTFILE,EXIST,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
     * OUTFILE,LOG
      IF(OUTFILE)CLOSE(12)
      OUTFILE=.TRUE.
      WRITE(9,80)'OUTPUT FILE NAME'
      IF(LOG)WRITE(1,80)'OUTPUT FILE NAME'
   80 FORMAT(A)
 6001 WRITE(9,80)'?  '
      READ(9,80)FILENAME
      INQUIRE(FILE=FILENAME,EXIST=EXIST)
      IF(.NOT.EXIST)GO TO 6005
      WRITE(9,80)TRIM(FILENAME)//' ALREADY EXISTS.  PLEASE CHOOSE ANOTHE
     *R NAME'
      GO TO 6001
 6005 OPEN(12,FILE=FILENAME)
      IF(LOG)WRITE(1,80)FILENAME
      RETURN
      END
      SUBROUTINE SEARCH(NEW,LIST,N,IRES,IPT)
C     RETURN 0 IF PATTERN IS IN LIST, ELSE RETURN POINTER TO NEXT HIGHER
      INTEGER LIST(2,N)
      IL=1
      IH=N
    1 I=(IL+IH)/2
      IF(NEW-LIST(2,I))5,10,15
    5 IH=I-1
      IF(IH.GE.IL)GO TO 1
      IRES=1
      IPT=I
      RETURN
   10 IRES=0
      IPT=I
      RETURN
   15 IL=I+1
      IF(IL.LE.IH)GO TO 1
      IRES=1
      IPT=I+1
      RETURN
      END
      INTEGER FUNCTION INTER(IN1X,IN2X)
      REAL*8 FREQ(2000)
      INTEGER INPAT(2000),BACKUPLIST(2,2000)
```

```
      LOGICAL OUTFILE,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NI,
     * OUTFILE,LOG
      IN1=IN1X
      IN2=IN2X
      IBIT=ISTART
      INTER=0
    1 IF(IN1.LT.IBIT)GO TO 5
      IN1=IN1-IBIT
      IF(IN2.LT.IBIT)GO TO 10
      IN2=IN2-IBIT
      INTER=INTER+IBIT
      GO TO 10
    5 IF(IN2.LT.IBIT)GO TO 10
      IN2=IN2-IBIT
   10 IBIT=IBIT/2
      IF(IBIT.GT.0)GO TO 1
      RETURN
      END
      SUBROUTINE MAP(LIST,NPATPA,NI)
C     SUBROUTINE TO READ A SET OF PATTERNS, SORT BY GRADE, DETERMINE EDGES
C     IN LATTICE, AND DERIVE MINIMAL ITEM-ABILITY REQUIREMENTS UNDER
C     ANTICHAIN POSET.
C     PROGRAM LIMITED TO 300 PATTERNS (NODES), EACH UP TO 30 ELEMENTS LONG.
C     DATA STRUCTURE:  NODE(1,I) IS GRADE, NODE(2-31,I) IS PATTERN,
C                NODE(32,I) IS SUCCESSOR COUNT, NODE(33,I) IS PTR TO EDGE
C                NODE(34,I) IS LABEL PASSED FROM CALLING PROGRAM IN LIST(1,J)
C                EDGE IS LIST OF SEQUENCE NUMBERS OF SUCCESSORS
C                GRADE(1,I) IS COUNT OF NODES OF GRADE I-1, GRADE(2,I)
C                IS SEQUENCE NUMBER OF FIRST NODE OF GRADE I-1
C                JIE(I) IS SEQUENCE NUMBER OF JIE
C                LABEL(I) IS PTR TO LABEL OF SKILL ASSOCIATED WITH JIE(I)
C                WORK(1,I) IS GRADE OF JIE, WORK(2,I) IS NEGATIVE OF
C                SEQUENCE NUMBER OF JIE, WORK(3,I) IS SEQUENCE NUMBER OF
C                LABEL (AFTER SORT)
C                ITEM(1,I) IS SKILL COUNT FOR ITEM I, ITEM(2,I) IS PTR
C                INTO REQS
C                REQS IS LIST OF ITEM SKILL REQUIREMENTS
C                SUC IS WORKING VECTOR USED TO LOCATE SUCCESSORS
      REAL*8 FREQ(2000)
      INTEGER NODE(34,500),EDGE(5000),JIE(50),WORK(3,50),ITEM(2,30),
     * GRADE(2,31),REQS(3000),LABEL(50),LIST(2,NPAT),IN(30)
      CHARACTER*1 LLIST(50),BLANK,X
      CHARACTER*80 FORMAT
      LOGICAL SUC(500),OUTFILE,LOG
      INTEGER INPAT(2000),BACKUPLIST(2,2000)
      CHARACTER*80 FILENAME
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NIX,
     * OUTFILE,LOG
      DATA BLANK/' '/,LLIST/'A','B','C','D','E','F','G','H','I','J','K',
     *'L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','a',
     * 'b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q',
     * 'r','s','t','u','v','w','x'/,GRADE/62*0/,MAXGRADE/0/
      NPAT=NPATPA
C     CHECK FOR 'NULL' AND 'FULL' NODES--FIRST 'NULL'
      DO 201 I=1,NI
  201 IN(I)=0
      J=IPACK(IN,NI)
```

BEST COPY AVAILABLE

```
      DO 202 I=1,NPAT
      IF(LIST(2,I).EQ.J)GO TO 205
  202 CONTINUE
      WRITE(9,80)'LIST AUGMENTED WITH ''NULL'''
      IF(LOG)WRITE(1,80)'LIST AUGMENTED WITH ''NULL'''
      NPAT=NPAT+1
      LIST(2,NPAT)=J
      LIST(1,NPAT)=0
C     NEXT 'FULL'
  205 DO 206 I=1,NI
  206 IN(I)=1
      J=IPACK(IN,NI)
      DO 208 I=1,NPAT
      IF(LIST(2,I).EQ.J)GO TO 210
  208 CONTINUE
      WRITE(9,80)'LIST AUGMENTED WITH ''FULL'''
      IF(LOG)WRITE(1,80)'LIST AUGMENTED WITH ''FULL'''
      NPAT=NPAT+1
      LIST(2,NPAT)=J
      LIST(1,NPAT)=0
   80 FORMAT(A)
  210 NI1=NI+1
      DO 10 IPAT=1,NPAT
      NODE(34,IPAT)=LIST(1,IPAT)
      CALL UNPACK(LIST(2,IPAT),NODE(2,IPAT),NI)
      NODE(1,IPAT)=0
      DO 4 I=2,NI1
    4 NODE(1,IPAT)=NODE(1,IPAT)+NODE(I,IPAT)
      NODE(32,IPAT)=0
      J=NODE(1,IPAT)
      GRADE(1,J+1)=GRADE(1,J+1)+1
      IF(J.GT.MAXGRADE)MAXGRADE=J
   10 CONTINUE
      MAXG1=MAXGRADE+1
C     SORT BY GRADE AND BY PATTERN WITHIN GRADE;
C     FINISH BUILDING GRADE TABLE
      DO 9 I=1,NPAT
      DO 9 J=2,NI1
    9 NODE(J,I)=-NODE(J,I)
      CALL SORT(NODE,NPAT,34,34)
      DO 11 I=1,NPAT
      DO 11 J=2,NI1
   11 NODE(J,I)=-NODE(J,I)
      GRADE(2,1)=1
      DO 12 I=2,MAXG1
   12 GRADE(2,I)=GRADE(1,I-1)+GRADE(2,I-1)
C     FIND SUCCESSORS
      IPT=0
      NODE(32,NPAT)=0
      DO 20 IPAT=1,NPAT
      IGD=NODE(1,IPAT)
      IF(IGD.EQ.MAXGRADE.AND.IPAT.NE.NPAT)GO TO 21
      IF(IGD.NE.MAXGRADE.AND.IPAT.EQ.NPAT)GO TO 21
      IF(IPAT.EQ.NPAT)GO TO 22
      IFND=IGD+2
      DO 13 IFND=IFND,MAXG1
      IF(GRADE(1,IFND).EQ.0)GO TO 13
      NSTART=GRADE(2,IFND)
```

```
      GO TO 131
   13 CONTINUE
      WRITE(9,80)'SERIOUS ERROR:  FELL OUT OF LOOP AT 13'
      IF(LOG)WRITE(1,80)'SERIOUS ERROR:  FELL OUT OF LOOP AT 13'
      GO TO 21
  131 NODE(32,IPAT)=0
      NODE(33,IPAT)=IPT+1
      DO 16 ICHEK=NSTART,NPAT
      DO 14 I=2,NI1
      IF(NODE(I,IPAT).GT.NODE(I,ICHEK))GO TO 15
   14 CONTINUE
      SUC(ICHEK)=.TRUE.
      GO TO 16
   15 SUC(ICHEK)=.FALSE.
   16 CONTINUE
      DO 18 ICHEK=NSTART,NPAT
      IF(.NOT.SUC(ICHEK))GO TO 18
      IF(ICHEK.EQ.NPAT)GO TO 19
      IST=ICHEK+1
      DO 17 ICK1=IST,NPAT
      IF(.NOT.SUC(ICK1))GO TO 17
      DO 161 I=2,NI1
      IF(NODE(I,ICHEK).GT.NODE(I,ICK1))GO TO 17
  161 CONTINUE
      SUC(ICK1)=.FALSE.
   17 CONTINUE
   18 CONTINUE
   19 DO 191 ICHEK=NSTART,NPAT
      IF(.NOT.SUC(ICHEK))GO TO 191
      NODE(32,IPAT)=NODE(32,IPAT)+1
      IPT=IPT+1
      EDGE(IPT)=ICHEK
  191 CONTINUE
   20 CONTINUE
      WRITE(9,80)'PROBLEM--FELL THROUGH LOOP AFTER 20'
      IF(LOG)WRITE(1,80)'PROBLEM--FELL THROUGH LOOP AFTER 20'
   21 WRITE(9,80)'SERIOUS ERROR:  NODE SET IS NOT A GRADED LATTICE'
      IF(LOG)WRITE(1,80)'SERIOUS ERROR:  NODE SET IS NOT A GRADED'//
     * ' LATTICE'
      WRITE(9,83)((NODE(I,J),I=1,34),J=1,NPAT)
      IF(LOG)WRITE(1,83)((NODE(I,J),I=1,34),J=1,NPAT)
   83 FORMAT(I4,30I1,3I3)
      WRITE(9,84)GRADE
      IF(LOG)WRITE(1,34)GRADE
   84 FORMAT((5(2I5,5X)))
      READ(9,80)X
      STOP
C     NORMAL EXIT FROM LOOP (ONLY NODE FOR WHICH THERE ARE ZERO NODES
C     OF NEXT HIGHER GRADE IS SINGLE NODE OF HIGHEST GRADE)
C     NEXT MUST FIND JIE'S AND ASSIGN LABELS TO CORRESPONDING ABILITIES
   22 NJIE=0
      DO 25 IPAT=1,NPAT
      IF(NODE(32,IPAT).GT.1)GO TO 25
      IF(IPAT.EQ.NPAT)GO TO 26
      IF(NODE(32,IPAT).EQ.1)GO TO 23
      WRITE(9,80)'SERIOUS PROBLEM AT LABEL 22+5'
      IF(LOG)WRITE(1,80)'SERIOUS PROBLEM AT LABEL 22+5'
      GO TO 21
```

```
   23 NJIE=NJIE+1
      JIE(NJIE)=IPAT
      WORK(1,NJIE)=NODE(1,IPAT)
   25 CONTINUE
      WRITE(9,80)'ERROR:  FELL THROUGH LOOP TO 25'
      IF(LOG)WRITE(1,80)'ERROR:  FELL THROUGH LOOP TO 25'
      GO TO 21
   26 WRITE(9,85)NJIE
      IF(LOG)WRITE(1,85)NJIE
   85 FORMAT(I5,' ABILITIES REQUIRED')
      DO 30 I=1,NJIE
      WORK(2,I)=-I
   30 WORK(3,I)=I
      CALL SORT(WORK,NJIE,3,3)
      DO 32 I=1,NJIE
      J=WORK(3,I)
   32 LABEL(I)=J
C     FIND SKILL REQUIREMENTS FOR EACH ITEM
      IPT=0
      DO 35 IT=1,NI
      IT1=IT+1
      ITEM(1,IT)=0
      ITEM(2,IT)=IPT+1
      DO 35 J=1,NJIE
      JSEQ=JIE(J)
      IF(NODE(IT1,JSEQ).EQ.1)GO TO 35
      ITEM(1,IT)=ITEM(1,IT)+1
      IPT=IPT+1
      REQS(IPT)=LABEL(J)
   35 CONTINUE
C     HERE TO OUTPUT LATTICE AND ITEM-ABILITY MAP
      WRITE(9,80)' '
      IF(LOG)WRITE(1,80)' '
      LINES=2
      ENCODE(21,86,FORMAT)NI-1,NI+14
   86 FORMAT('(3I4,',I2,'I1,(T',I2,',10I4))')
      DO 40 IPAT=1,NPAT
      J1=NODE(33,IPAT)
      J2=J1+NODE(32,IPAT)-1
      WRITE(9,FORMAT)NODE(34,IPAT),(NODE(I,IPAT),I=1,NI1),
     * (NODE(34,EDGE(I)),I=J1,J2)
      IF(LOG)WRITE(1,FORMAT)NODE(34,IPAT),(NODE(I,IPAT),I=1,NI1),
     * (NODE(34,EDGE(I)),I=J1,J2)
      LINES=LINES+(J2-J1+9)/10
      IF(J1.GE.J2)LINES=LINES+1
      IF(LINES.LE.25)GO TO 40
      WRITE(9,80)'(TYPE <CR> TO CONTINUE)'
      READ(9,80)X
      LINES=0
   40 CONTINUE
      WRITE(9,80)'(TYPE <CR> TO CONTINUE)'
      READ(9,80)X
      WRITE(9,80)' '
      WRITE(9,80)'ITEM-ABILITY REQUIREMENTS ASSUMING ANTICHAIN POSET'
      IF(LOG)
     *WRITE(1,80)'ITEM-ABILITY REQUIREMENTS ASSUMING ANTICHAIN POSET'
      DO 45 IT=1,NI
      J1=ITEM(2,IT)
```

```
      J2=ITEM(1,IT)
      IF(J2.LT.2)GO TO 44
      CALL SORT(REQS(J1),J2,1,1)
   44 J2=J1+ITEM(1,IT)-1
      WRITE(9,88)IT,(LLIST(REQS(I)),I=J1,J2)
      IF(LOG)WRITE(1,88)IT,(LLIST(REQS(I)),I=J1,J2)
   45 CONTINUE
   88 FORMAT(I4,(T10,30A1))
      RETURN
      END
      INTEGER FUNCTION IPACK(IN,NI)
      INTEGER IN(NI)
      IPACK=0
      DO 5 I=1,NI
    5 IPACK=2*IPACK+IN(I)
      RETURN
      END
      SUBROUTINE UNPACK(J1,IN,NI)
      REAL*8 FREQ(2000)
      INTEGER INPAT(2000),BACKUPLIST(2,2000)
      INTEGER IN(NI)
      LOGICAL OUTFILE,LOG
      COMMON/START/FREQ,INPAT,BACKUPLIST,MAXIN,ISTART,ISER,NN,
     * OUTFILE,LOG
      I=1
      J=J1
      IBIT=ISTART
    1 IF(J-IBIT)5,10,10
    5 IN(I)=0
      GO TO 15
   10 IN(I)=1
      J=J-IBIT
   15 I=I+1
      IBIT=IBIT/2
      IF(IBIT.GT.0)GO TO 1
      RETURN
      END
      SUBROUTINE SORT(IBUF,N,LEN,ISKIP)
C     GENERAL-PURPOSE SORTING SUBROUTINE, USING 'QUICK SORT' ALGORITHM
C     IBUF IS VECTOR TO BE SORTED, N IS NUMBER OF ELEMENTS IN VECTOR,
C     LEN IS NUMBER OF WORDS IN EACH ELEMENT, AND ISKIP (≥ LEN) IS NUMBER
C     OF WORDS SEPARATING THE BEGINNING OF SUCCESSIVE VECTOR ELEMENTS
C     LEN MUST BE LESS THAN OR EQUAL TO THE DIMENSION OF PIVOT
      INTEGER IBUF(ISKIP,N),LV(40),UV(40),PIVOT,P
      LV(1)=1 ; UV(1)=N ; P=1
      DO WHILE (P.GT.0)
      IF(LV(P).GE.UV(P)) THEN
        P=P-1
      ELSE
        I=LV(P)-1
      J=UV(P)
        PIVOT=J
    1 I=I+1
      IF(I.GE.J) GO TO 10
      DO 2 KX=1,LEN
      IF(IBUF(KX,I)-IBUF(KX,PIVOT))1,2,4
    2   CONTINUE
        GO TO 1
```

BEST COPY AVAILABLE

```
 4 J=J-1
   IF(J.LE.I) GO TO 10
   DO 6 KX=1,LEN
   IF(IBUF(KX,J)-IBUF(KX,PIVOT))7,6,4
 6   CONTINUE
     GO TO 4
 7 DO 8 KX=1,LEN
     L=IBUF(KX,I)
     IBUF(KX,I)=IBUF(KX,J)
 8     IBUF(KX,J)=L
   GO TO 1
10 DO 12 KX=1,LEN
     IF(IBUF(KX,I)-IBUF(KX,PIVOT))13,12,14
12 CONTINUE
13 IF(I.EQ.PIVOT)GO TO 16
   I=I+1
14 DO 15 KX=1,LEN
   L=IBUF(KX,I)
   IBUF(KX,I)=IBUF(KX,PIVOT)
15 IBUF(KX,PIVOT)=L
16 IF (I-LV(P) .LT. UV(P)-I)THEN
    LV(P+1)=LV(P)
    UV(P+1)=I-1
    LV(P)=I+1
   ELSE
    LV(P+1)=I+1
    UV(P+1)=UV(P)
    UV(P)=I-1
   END IF
   P=P+1
   END IF
   REPEAT
   RETURN
   END
```

```
(* :Title: Ability/Task Representations *)

(* :Author: Edward H. Haertel *)

(* :Summary:
This package implements a set of functions useful in
the analysis and representation of discrete ability and
task structures, as proposed by Haertel & Wiley (1993).
It extends the package
Combinatorica,  Copyright 1990-1993 by Steven S. Skiena
*)

(* :Discussion:
Comments should be directed to the author at
haertel@leland.stanford.edu or 415-725-1251.
*)

(* :Context: DiscreteMath`Combinatorica`
*)

(* :Package Version: 1.0, October 1994
*)
```

In[1]:=

```
<<DiscreteMath`Combinatorica`
```

In[2]:=

```
(*Define lower-case letters as corresponding text
strings for use as (atomic) ability labels *)

a="a";b="b";c="c";d="d";e="e";f="f";g="g";h="h";i="i";
j="j";k="k";l="l";m="m";n="n";o="o";p="p";q="q";r="r";
s="s";t="t";u="u";v="v";w="w";x="x";y="y";z="z";
```

In[6]:=

```
(* AList[AbilityCombinationList] returns a list of the
   primative abilities included in one or more ability
   combinations. *)

AList[acomb_List]:=
     Union[Flatten[Characters /@ Flatten[acomb] ] ]
```

```
General::spell1:
   Possible spelling error: new symbol name "AList"
     is similar to existing symbol "List".
```

BEST COPY AVAILABLE

In[8]:=

```
(* JIEs[DepList] returns a list of the join-irreducible
elements implied by a list of (pairwise) ability
dependencies.  The abilities list must be a list of
elements of the form {antecedent-ability-label,
dependent-ability-label} or {ability-label} or
ability-label, or any mixture.  The latter two forms
permit referencing of abilities that do not enter into
antecedent/postcedent relations with any other
abilities *)

JIEs[dep_List]:=JIE$Temp /; (JIE$Temp=
Module[{alist,n,dummyvertices,ttemp,utemp},
     (alist=AList[dep];n=Length[alist];
     dummyvertices=Table[{0,0},{n}];
     ttemp=Table[0,{n},{n}];
     Scan[(ttemp[[#[[1]],#[[2]] ]]=1)&,
          Map[{Position[alist,#[[1]]][[1,1]],
               Position[alist,#[[2]]][[1,1]]}&,
               Select[dep,(ListQ[#] && Length[#]==2)&]
          ],
     1];
     utemp=Edges[TransitiveClosure[
          Graph[ttemp,dummyvertices] ] ];
     If[AcyclicQ[Graph[utemp,dummyvertices],Directed],
        StringJoin /@
          Map[alist[[Flatten[Position[#,1],2] ]]&,
           Transpose[utemp+IdentityMatrix[n]]
          ],
        $Failed
     ]
)]; JIE$Temp =!= $Failed)

General::spell1:
   Possible spelling error: new symbol name "utemp"
     is similar to existing symbol "ttemp".
```


BEST COPY AVAILABLE

In[10]:=

```
(* APatterns[JIEList] returns a list of all possible
ability patterns generated by the poset represented by
a list of join-irreducible elements (i.e., the nodes
of the distributive lattice having a given set of
join-irreducible elements).
*)

APatterns[jies_List]:=
Map[
     (Apply[StringJoin,#])&,
     Union[
          Map[Union,Characters /@
               ((Apply[StringJoin,#])& /@ Subsets[jies])
          ]
     ]
]

(* Here is an example of the use of JIEs and
   APatterns *)
JIEs[{h,{d,b},{b,a},{a,c},{g},{h,a}}]
```

```
{abdh, bd, abcdh, d, g, h}
```

```
APatterns[%]
```

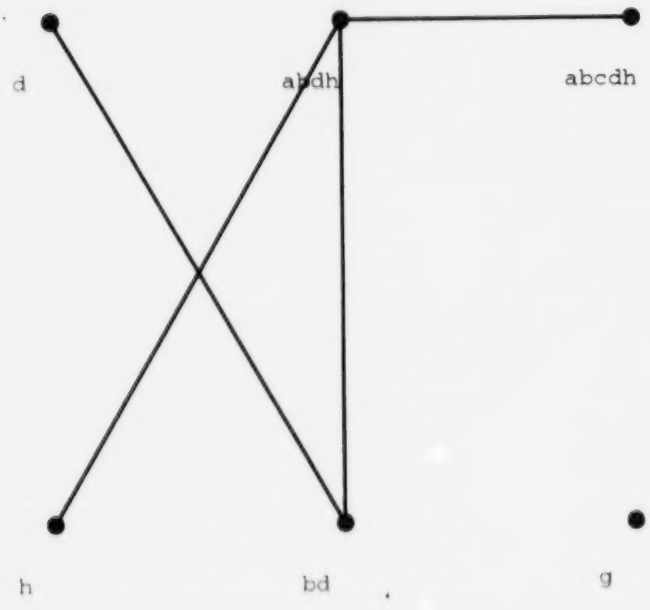
```
{, d, g, h, bd, dg, dh, gh, bdg, bdh, dgh, abdh, bdgh,
  abcdh, abdgh, abcdgh}
```

In[12]:=

```
(* Poset[JIEList] returns a graph of the ability poset
(partially ordered set) corresponding to a list of
join-irreducible elements.  The graph is a Hasse
diagram presented left-to-right, with independent
components left-justified. *)

Poset[JIE_List]:=
Module[{n=Length[JIE],i,j,rawgraph},
     rawgraph=HasseDiagram[
       Graph[
          Table[
               If [And @@ (StringMatchQ[JIE[[j]],
                    "*"<>#<>"*"]& /@ Characters[JIE[[i]] ]),
                    1, 0
               ],
          {i,n},{j,n}],
       Table[{0,0},{n}] ] ];
     Graph[First[rawgraph],
           {#[[2]],-#[[1]]}& /@ Last[rawgraph]
     ]
]
```

BEST COPY AVAILABLE

```
(* Here is an illustration of the use of Poset
   Note that it does not work exactly as intended;
   In general, the problem of finding pleasing
   planar representations of graphs is extremely
   difficult.  The embeddings created by Poset
   should be taken as first approximations. *)
ShowLabeledGraph[Poset[
{"abdh", "bd", "abcdh", "d", "g", "h"}],
{"abdh", "bd", "abcdh", "d", "g", "h"}]
```



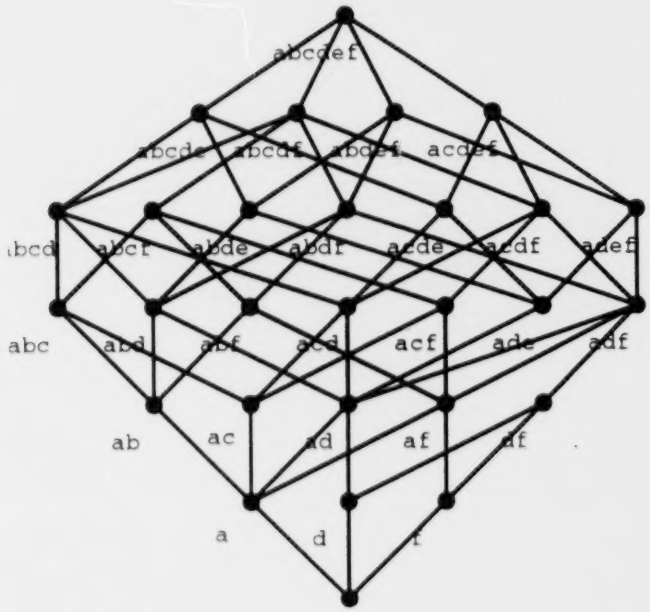
-Graphics-

In[14]:=

```
(* The following is a demonstration of the use of the
   above routines together with Combinatorica routines
   to construct a complete Hasse diagram corresponding
   to the full set of ability-states implied by a given
   set of pairwise dependencies (and single elements).

   Note that the function pathQ tests for a subset
   relation between any pair of ability-state labels. *)

pathQ[x_String,y_String]:=
   Apply[And,Map[StringMatchQ[y,"*"<>#<>"*"]&,
   Characters[x]]]

AStateEmbed[demolist_List]:=
Module[{abilitylist},
ShowLabeledGraph[
     HasseDiagram[
          MakeGraph[
               nodelist=APatterns[
                    JIEs[demolist]
               ],
               pathQ
          ]
     ],
     nodelist
] ]

demolist={{a,b},{a,c},{d,e},{a,e},f}

AStateEmbed[demolist]
```

Out[19]=

```
{{a, b}, {a, c}, {d, e}, {a, e}, f}
```



Out[20]=

```
-Graphics-
```

In[21]:=

```
(* Here are JIEs, Posets, and Ability State Lattices
   for the five nonisomorphic structures on 3 binary
   abilities.  (Structures are specified initially in
   terms of pairwise dependencies between abilities.)
*)

structures={{a,b,c},
            {{a,b},c},
            {{a,b},{a,c}},
            {{a,c},{b,c}},
            {{a,b},{b,c}}}

TableForm[jiesets=JIEs[#]& /@ structures]

ShowLabeledGraph[Poset[#],#]& /@ jiesets

AStateEmbed[#]& /@ structures
```

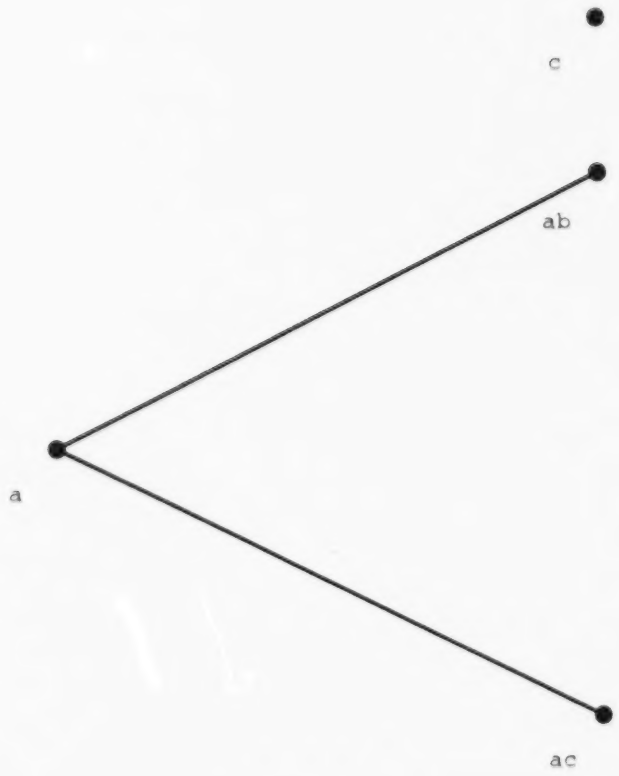
Out[23]=

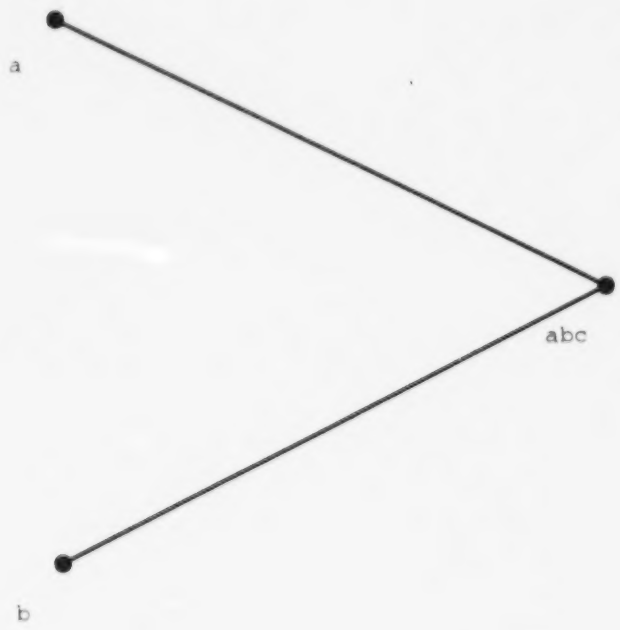
```
{{a, b, c}, {{a, b}, c}, {{a, b}, {a, c}},
  {{a, c}, {b, c}}, {{a, b}, {b, c}}}
```

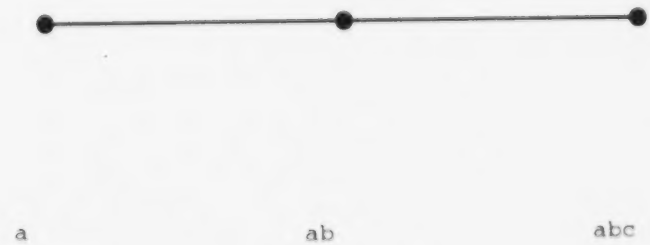
Out[24]//TableForm=

a	b	c
a	ab	c
a	ab	ac
a	b	abc
a	ab	abc

ab
a

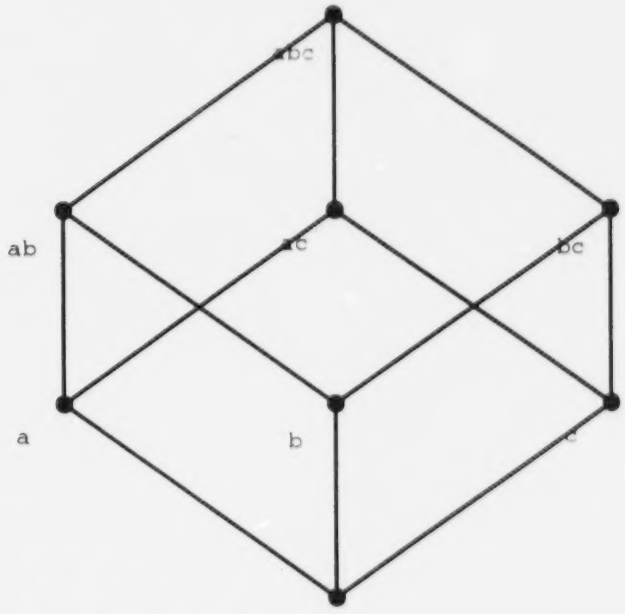

c
ab
a
ac

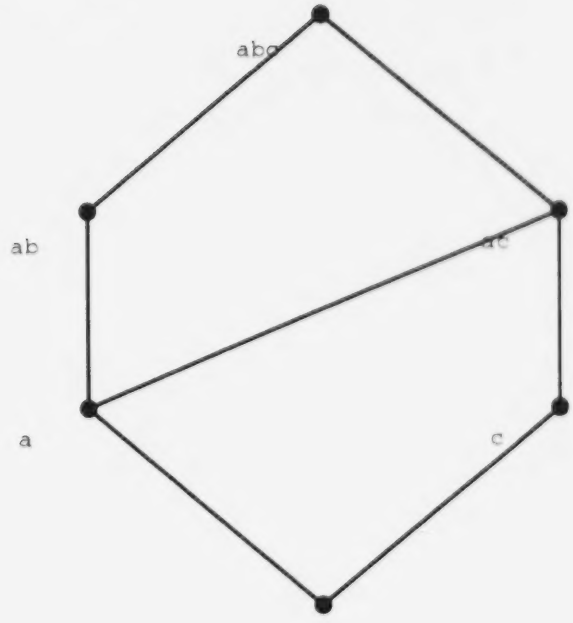


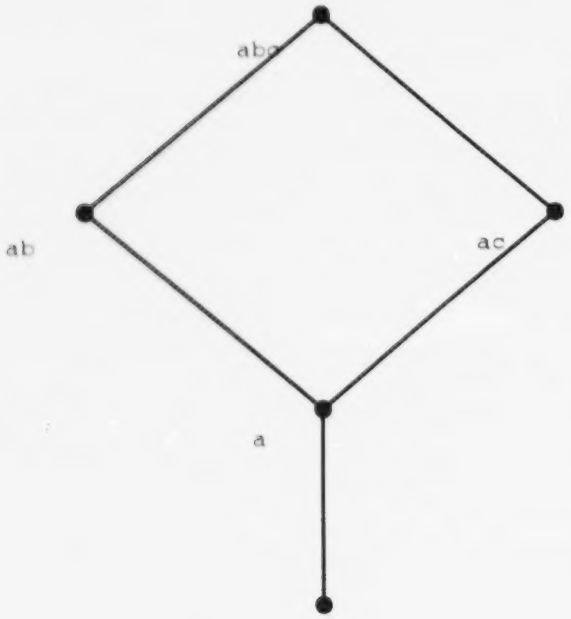


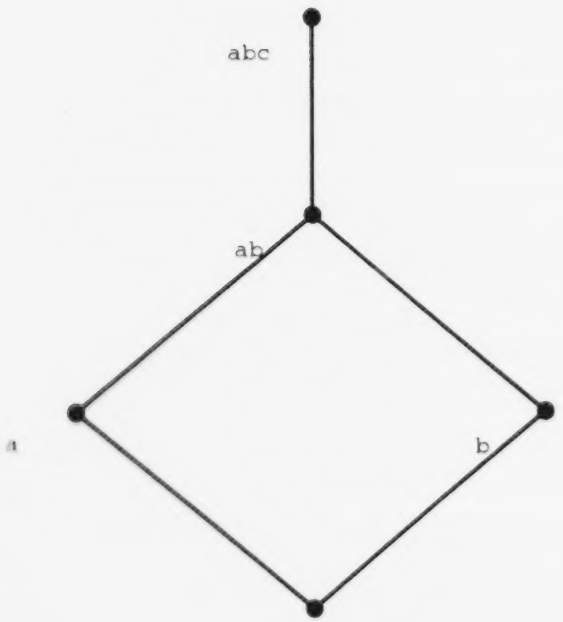
Out[25]=

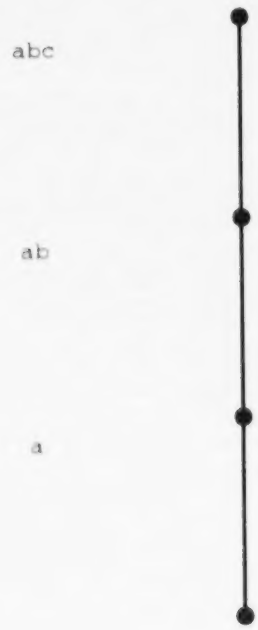
```
{-Graphics-, -Graphics-, -Graphics-, -Graphics-, -Graphics-}
```


abc
ab
ac
bc
a
b
c


abc
ab
ac
a
c


abc
ab
ac
a


abc
ab
a
b



Out[26]=

(-Graphics-, -Graphics-, -Graphics-, -Graphics-, -Graphics-)